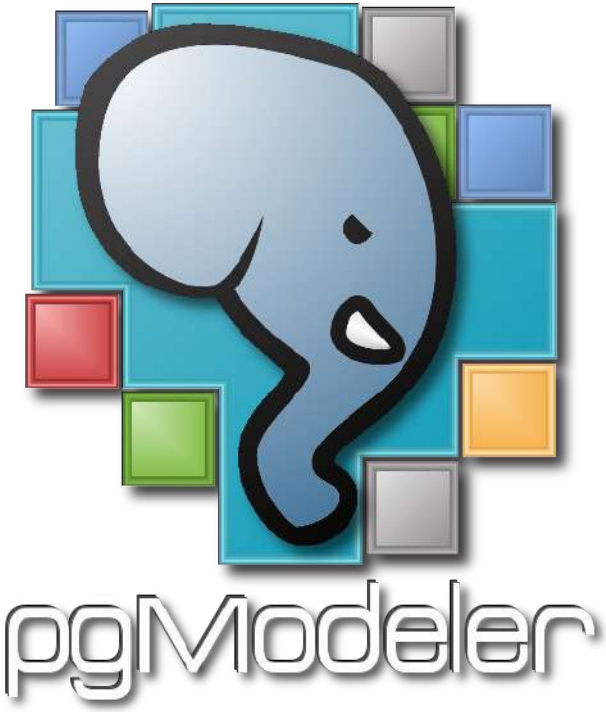


pgModeler: muito mais que um modelador de bancos de dados PostgreSQL

PGConf.Brasil 2018

Raphael Araújo e Silva

- Bacharel em Ciência da Computação pela Universidade Federal do Tocantins;
- Há 12 anos atuando como Desenvolvedor de Software na Assembleia Legislativa do Estado do Tocantins;
- Apaixonado pela programação, teoria de compiladores, computação gráfica, processamento de imagens e entusiasta da área de banco de dados.

raphael@pgmodeler.io

/rkhaotix /raphasilva /pgmodeler /pgmodeler

Como tudo começou...

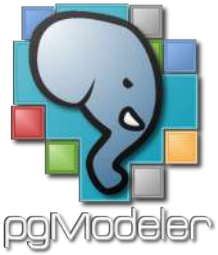
- Necessidade de documentação para os sistemas que desenvolvia;
- Escassez de ferramentas livres ou baratas para modelagem de dados e geração de código;
- As poucas ferramentas que encontrava à época não atendiam por completo às demandas.

#chateado :(

Hello, world!

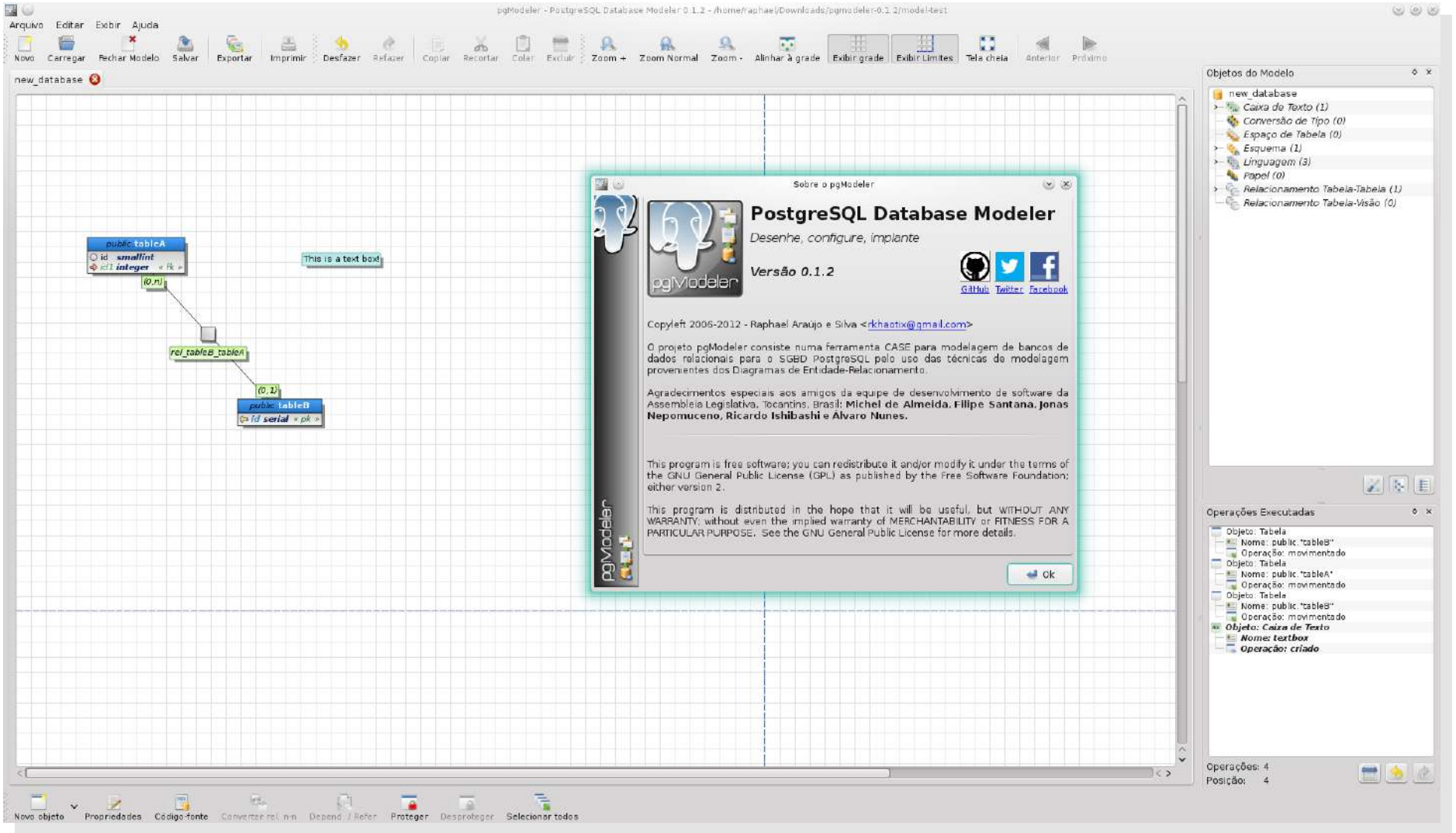
- Em 2006 nascia o projeto PostgreSQL Database Modeler ou **pgsqlDBM**. Anos depois alterado para **pgModeler**;

- Concebido sem intenção de divulgação, porém, em 2012 foi liberado sob licença **GPL2** e posteriormente atualizado para **GPL3**.



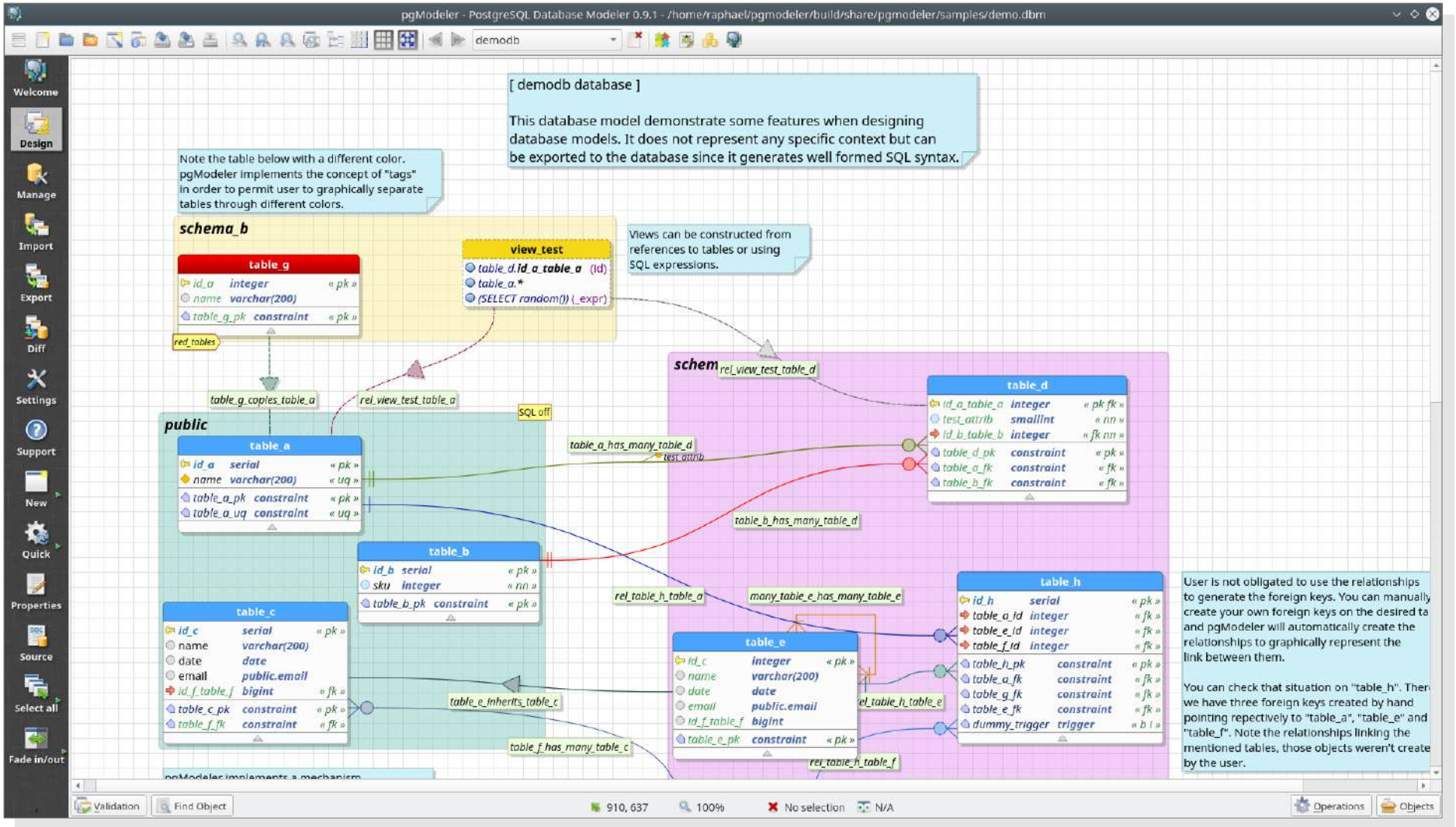
Hello, world!

pgModeler 0.1.2 (Agosto, 2012)



Hello, world!

pgModeler 0.9.1 (Maio, 2018)



Do que ele é feito?



- Implementado em sua maior parte em **ISO C++03** (migrando para **ISO C++11/14**) e utilizando o framework **Qt**;
- Compatível com as 3 principais plataformas desktop: **Linux**, **Windows** e **macOS**;
- Encapsula funções da **libpq** para operações em banco de dados e da **libxml2** para geração de diversos tipos de arquivos;
- Implementa uma micro linguagem de templating para geração de código SQL e XML de forma dinâmica.

Do que ele é capaz?

- Modelagem de dados utilizando conceitos clássicos de entidade-relacionamento;
- Geração de código SQL: modele uma vez e exporte para diferentes versões do PostgreSQL (9.x e 10.x);
- Validação da modelagem de dados com aplicação de correções semi-automáticas;
- Engenharia reversa de BD consultando os catálogos do sistema: **pg_catalog.*** e **information_schema.***;

Do que ele é capaz?

- Comparação de banco de dados e geração de código SQL para sincronização de entidades;
- Gerenciamento de banco de dados e manipulação dos dados de tabelas de forma facilitada;
- Possui uma versão CLI (linha de comando) que encapsula algumas das principais funcionalidades;
- Extras: suporte a tipos de dados geoespaciais (PostGIS), suporte a plug-ins, mecanismo de segurança para resguardar o trabalho em caso de bugs ou crashes, e muitos outros...

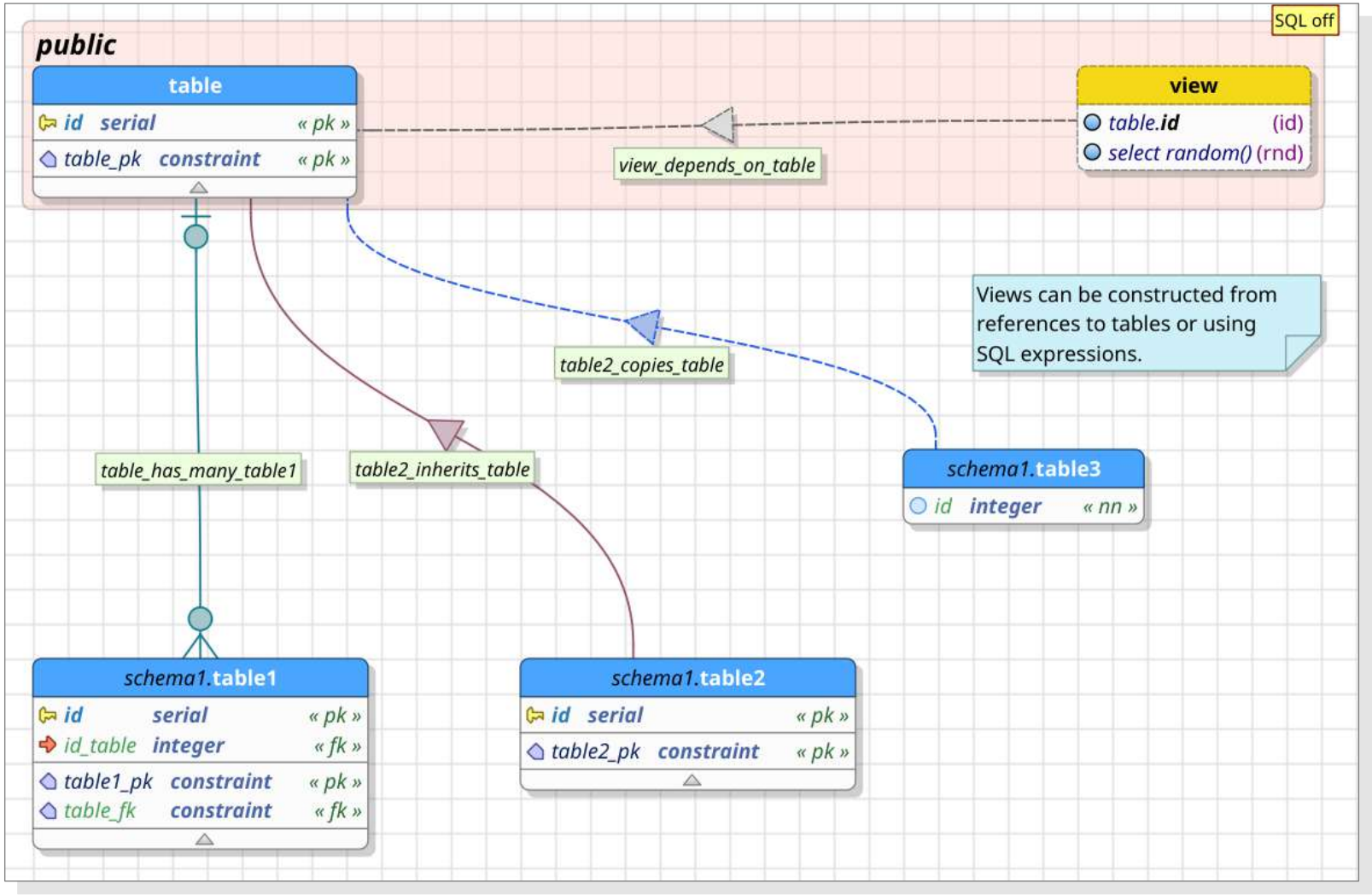
Modelagem de dados

Modelagem de dados

- Documentação, visão macro de um banco de dados, geração de código;
- Permite o agrupamento visual de objetos por esquemas ou tags;
- Modelos podem ser exportados para PNG ou SVG;
- Implementa um mecanismo de propagação automática de colunas e restrições;

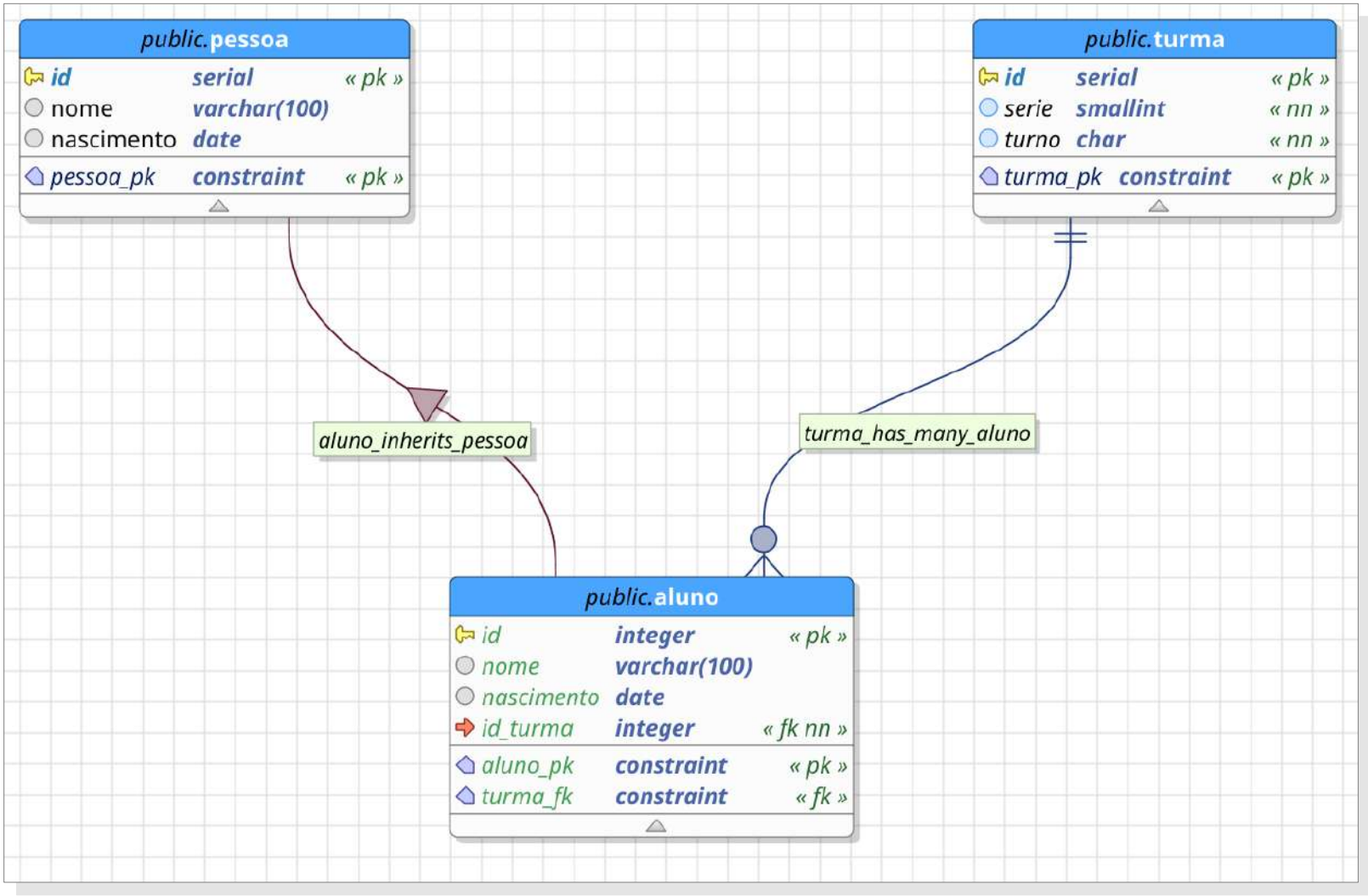
Modelagem de dados

- Entidades gráficas:



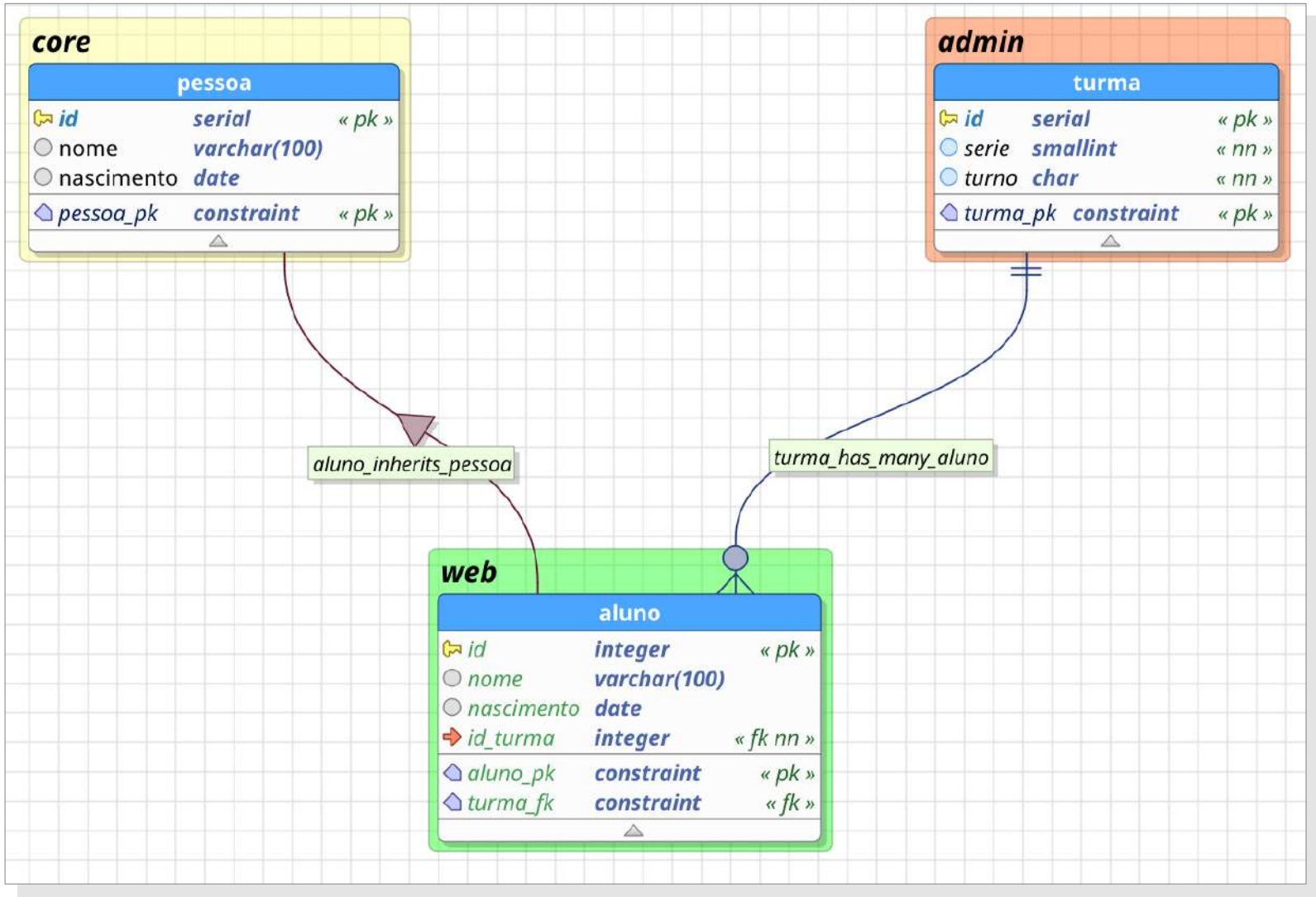
Modelagem de dados

- Propagação de colunas e restrições:



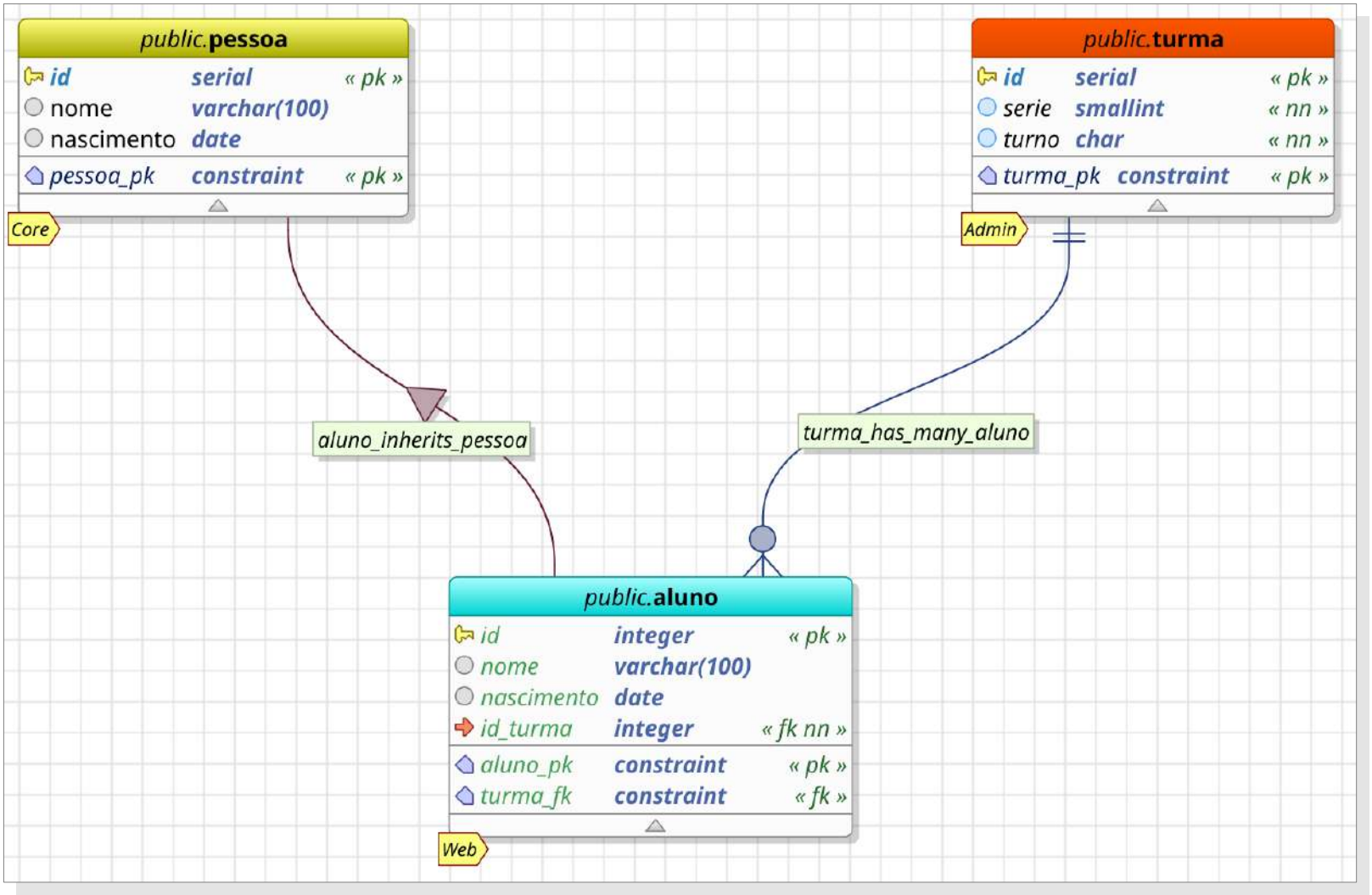
Modelagem de dados

- Agrupamento por esquemas:



Modelagem de dados

- Agrupamento por tags:



Geração de código SQL

Geração de código SQL

- É a função primordial da ferramenta;

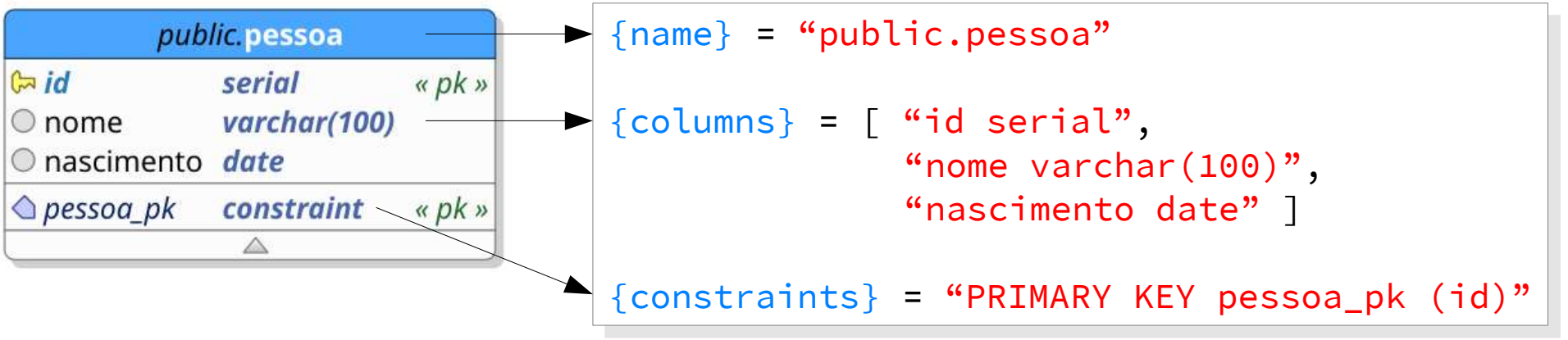
- Ocorre em 3 etapas:
    1. Extração dos atributos dos objetos;
    2. Parsing dos templates dos objetos;
    3. Conversão dos templates interpretados em código SQL;

Geração de código SQL

1. Extração dos atributos do objeto:

Geração de código SQL

1. Extração dos atributos do objeto:



Geração de código SQL

2. Parsing do template do objeto:

```
{name} = “public.pessoa”

{columns} = [ “id serial”,
              “nome varchar(100)”,
              “nascimento date” ]

{constraints} = “PRIMARY KEY
                   pessoa_pk (id)”
```

Geração de código SQL

2. Parsing do template do objeto:

```
{name} = “public.pessoa”

{columns} = [ “id serial”,
              “nome varchar(100)”,
              “nascimento date” ]

{constraints} = “PRIMARY KEY
                   pessoa_pk (id)”
```

```
# SQL definition for tables
[-- object: ] {name} [ | type: ] {sql-object} [ --] $br
[-- ] {drop}

[CREATE]

%if ({pgsql-ver} != "9.0") %and {unlogged} %then
     [ UNLOGGED]
%end

[ TABLE ] {name} ( $br

%if {columns} %then
     {columns}
%end

%if {constraints} %then
     {constraints}
%end

$br );

[-- ddl-end --] $br
```

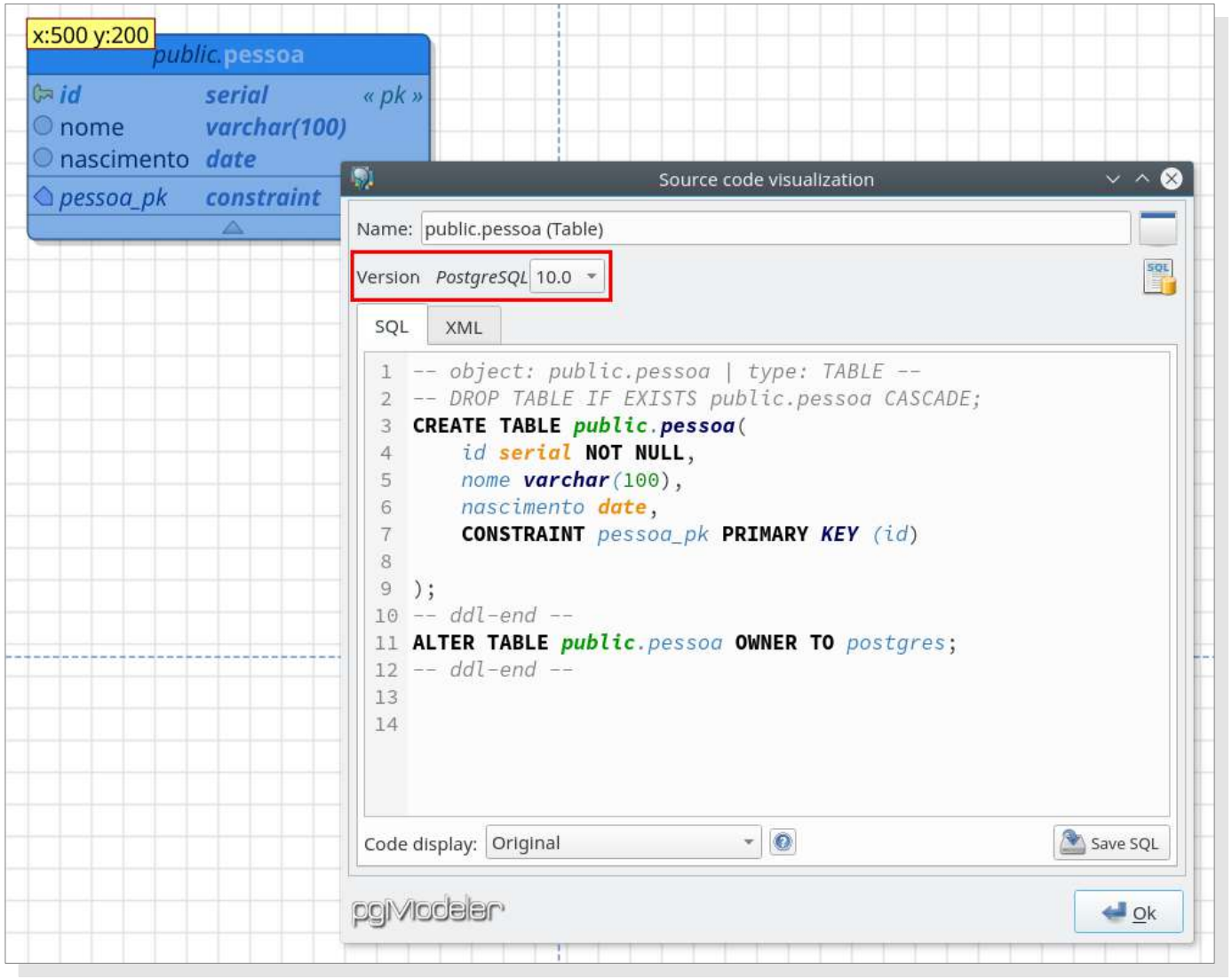
Geração de código SQL

3. Código SQL resultante:

```
# SQL definition for tables
[-- object: ] {name} [ | type: ] {sql-object} [ --] $br
[-- ] {drop}

[CREATE]

%if ({pgsql-ver} != "9.0") %and {unlogged} %then
     [ UNLOGGED]
%end

[ TABLE ] {name} ( $br

%if {columns} %then
     {columns}
%end

%if {constraints} %then
     {constraints}
%end

$br );

[-- ddl-end --] $br
```

```
-- object: public.pessoa | type: TABLE --
-- DROP TABLE IF EXISTS public.pessoa CASCADE;
CREATE TABLE public.pessoa(
    id serial NOT NULL,
    nome varchar(100),
    nascimento date,
    CONSTRAINT pessoa_pk PRIMARY KEY (id)
);
-- ddl-end --
```

Geração de código SQL



Engenharia reversa

Engenharia reversa

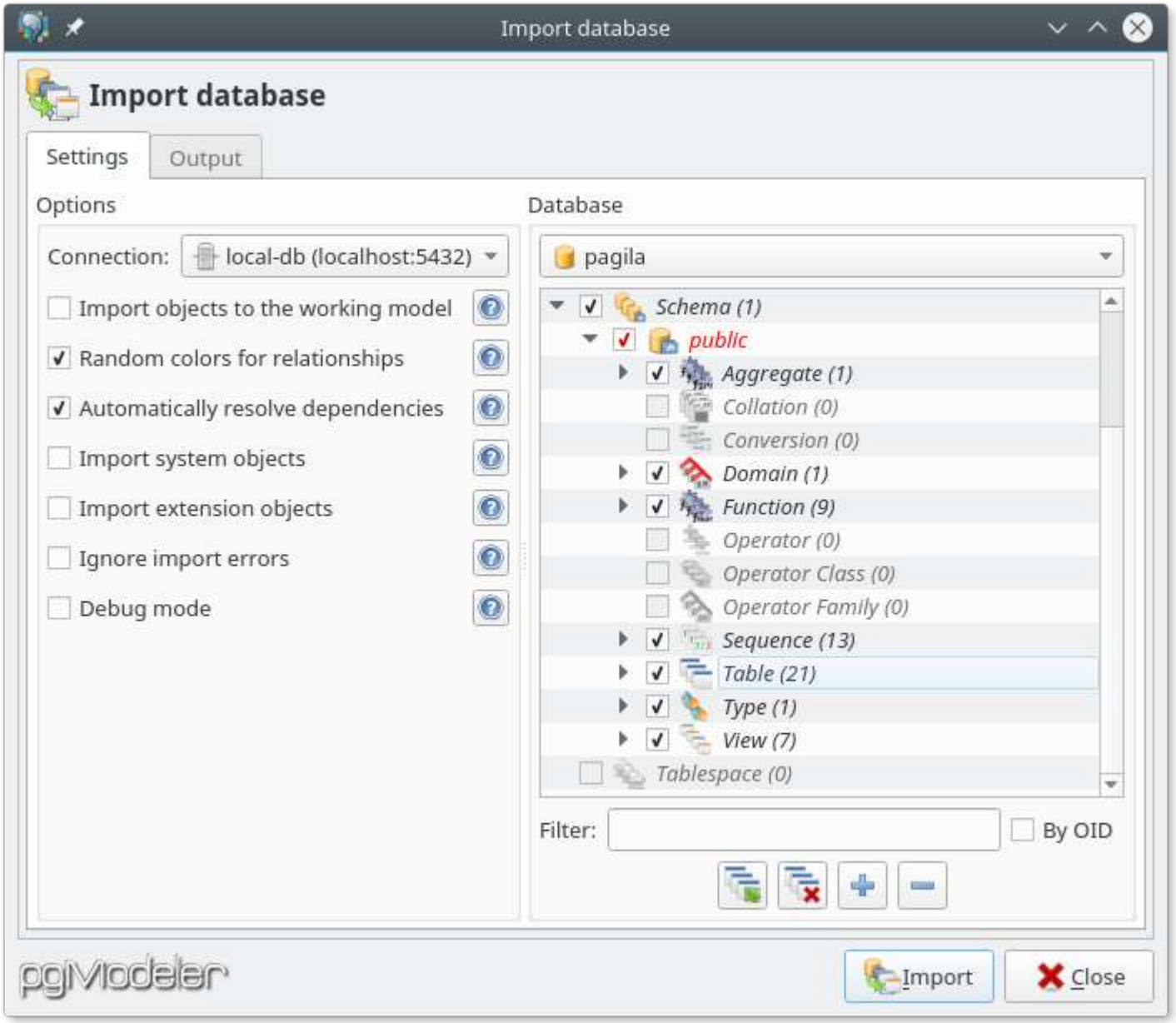
- Leitura dos catálogos do sistema para a reconstrução do modelo de banco de dados;

- Permite a importação de um banco de dados (ou frações deste) resolvendo dependências automaticamente;

- Executada em três etapas:

  1. Parsing do template e geração da query de catálogo;
  2. Geração de um código XML que representa o objeto do banco de dados;
  3. Interpretação do código XML criando o objeto no modelo.

Engenharia reversa



Engenharia reversa

1. Parsing do template e geração da query de catálogo:

```
[ SELECT tb.oid, tb.relname AS name, tb.relnamespace AS schema,
tb.relowner AS owner, tb.reltuples::int8 AS row_amount,
tb.reltablespace AS tablespace, tb.relacl AS permission,
relhasoids AS oids_bool, ]

%if ({pgsql-ver} == "9.0") %then
     [ FALSE AS unlogged_bool, ]
%else
     [ CASE relpersistence
         WHEN 'u' THEN TRUE
         ELSE FALSE
       END
       AS unlogged_bool, ]
%end

%if ({pgsql-ver} <f "9.5") %then
     [ FALSE AS rls_enabled_bool, ]
     [ FALSE AS rls_forced_bool, ]
%else
     [ tb.relrowsecurity AS rls_enabled_bool, ]
     [ tb.relforcerowsecurity AS rls_forced_bool, ]
%end

({comment}) [ AS comment ]

[ FROM pg_class AS tb
  LEFT JOIN pg_tables AS _tb1 ON _tb1.tablename=tb.relname
  WHERE tb.relkind='r' ]

%if {schema} %then [ AND _tb1.schemaname= ] '{schema}' %end
```

Engenharia reversa

1. Parsing do template e geração da query de catálogo:

```
SELECT tb.oid, tb.relname AS name, tb.relnamespace AS schema,
       tb.relowner AS owner, tb.reltuples::int8 AS row_amount,
       tb.reltablespace AS tablespace, tb.relacl AS permission,
       relhasoids AS oids_bool,
       CASE relpersistence WHEN 'u' THEN TRUE
       ELSE FALSE END AS unlogged_bool,
       tb.relrowsecurity AS rls_enabled_bool, tb.relforcerowsecurity AS rls_forced_bool,
       (SELECT array_agg(inhparent) AS parents FROM pg_inherits WHERE inhrelid = tb.oid),
       (SELECT description FROM pg_description WHERE objoid = tb.oid AND objsubid = 0) AS comment
FROM pg_class AS tb
LEFT JOIN pg_tables AS _tb1 ON _tb1.tablename=tb.relname
WHERE tb.relkind='r' AND
      tb.oid > 13018 AND
      (tb.oid NOT IN (13006,13007,13008,13009)) AND
      tb.oid IN (26551,26557,26565);
```

Engenharia reversa

2. Geração do código XML que representa o objeto:

```
<table name="pessoa">
    <schema name="public"/>
    <role name="postgres"/>
    <position x="500" y="200"/>
    <column name="id" not-null="true">
        <type name="serial" length="0"/>
    </column>
    <column name="nome">
        <type name="varchar" length="100"/>
    </column>
    <column name="nascimento">
        <type name="date" length="0"/>
    </column>
    <constraint name="pessoa_pk" type="pk-constr" table="public.pessoa">
        <columns names="id" ref-type="src-columns"/>
    </constraint>
</table>
```

Engenharia reversa

3. Geração do objeto a partir do código XML:

```
<table name="pessoa">
    <schema name="public"/>
    <role name="postgres"/>
    <position x="500" y="200"/>
    <column name="id" not-null="true">
        <type name="serial" length="0"/>
    </column>
    <column name="nome">
        <type name="varchar" length="100"/>
    </column>
    <column name="nascimento">
        <type name="date" length="0"/>
    </column>
    <constraint name="pessoa_pk" type="pk-constr" table="public.pessoa">
        <columns names="id" ref-type="src-columns"/>
    </constraint>
</table>
```

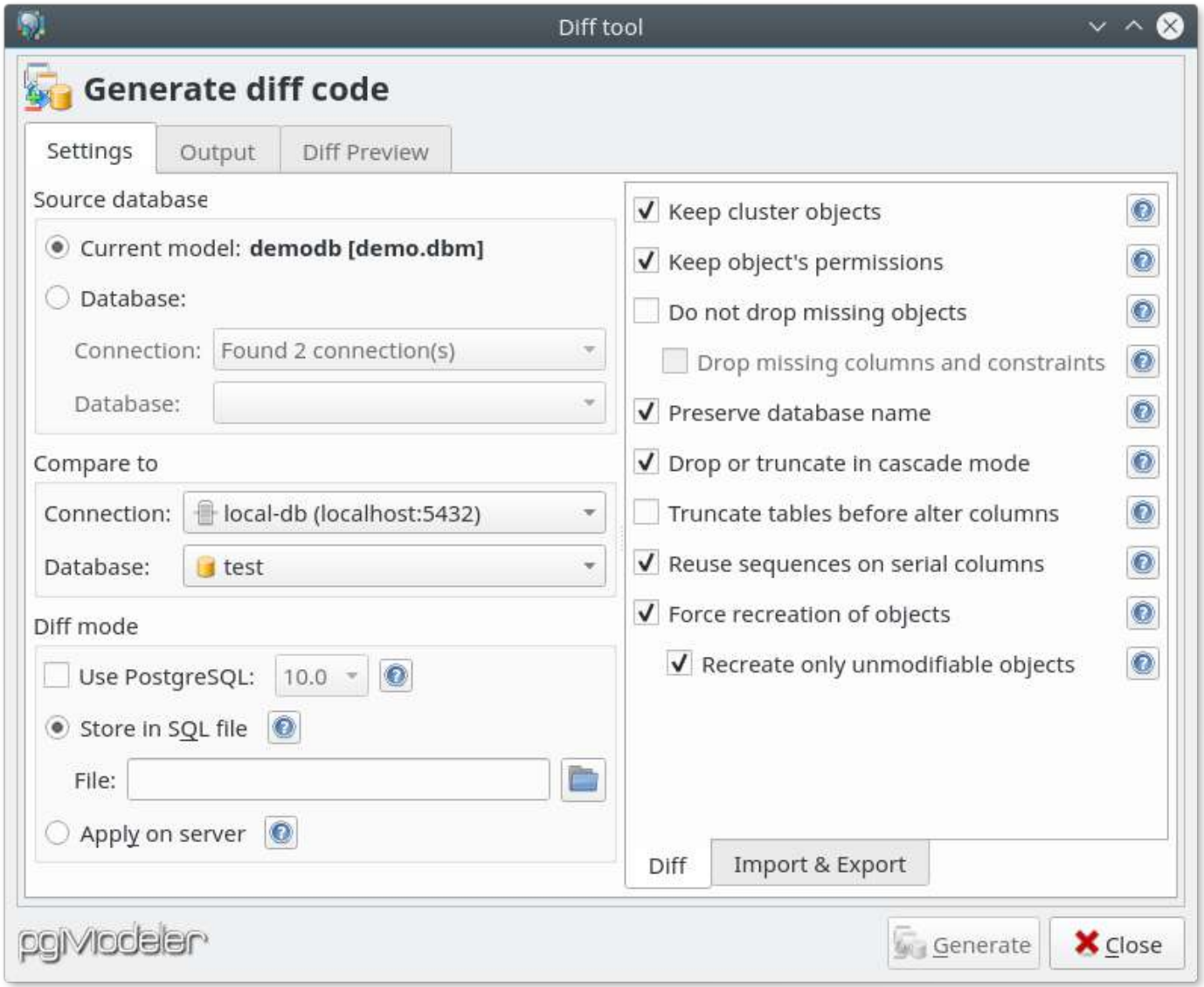
Comparação de BDs (diff)

Comparação de banco de dados

- Detecta diferenças entre um modelo e um banco de dados ou entre dois bancos de dados;

- Gera o código SQL capaz de sincronizar o banco de dados deixando-o semelhante ao modelo ou BD utilizado como entrada;

Comparação de banco de dados



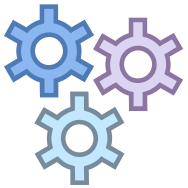
Comparação de banco de dados

dbtest@localhost

dbtest.dbm

Comparação de banco de dados

dbtest@localhost

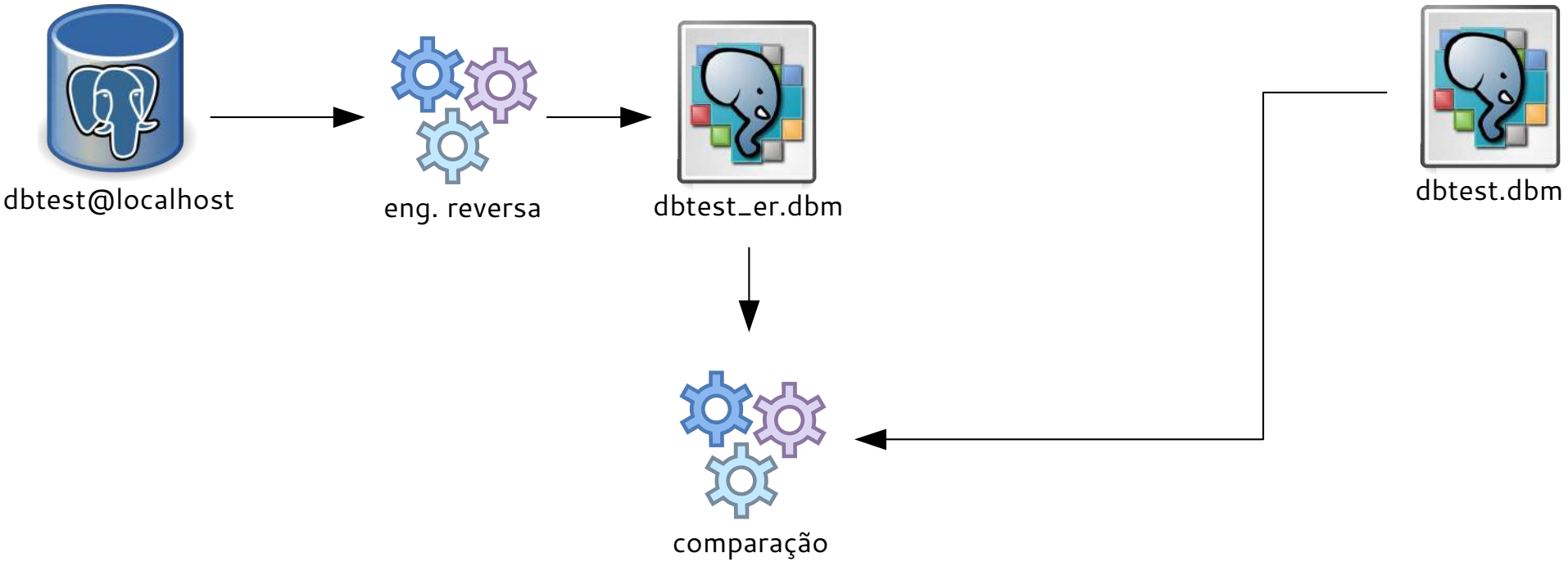


eng. reversa

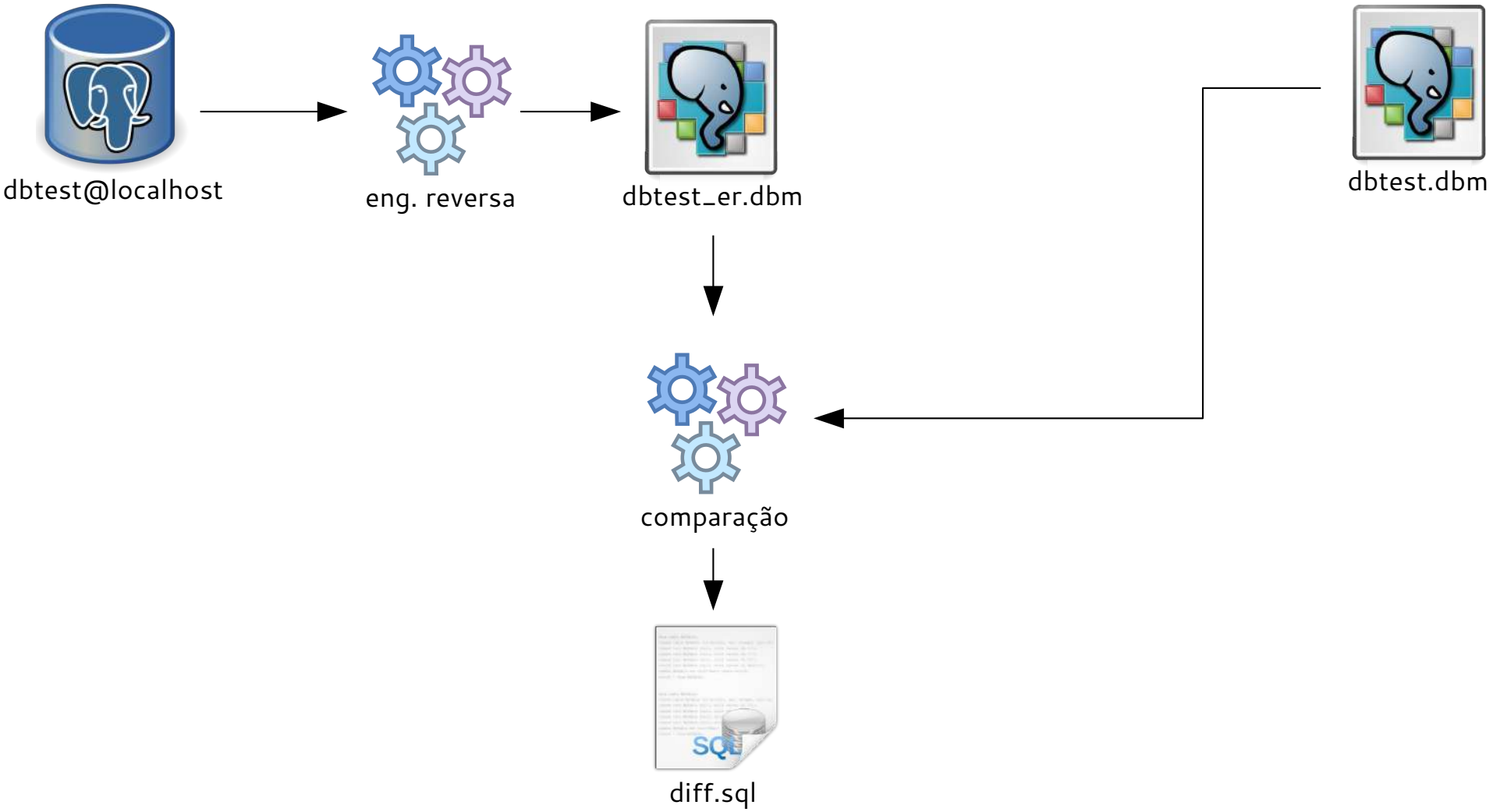
dbtest.dbm

Comparação de banco de dados

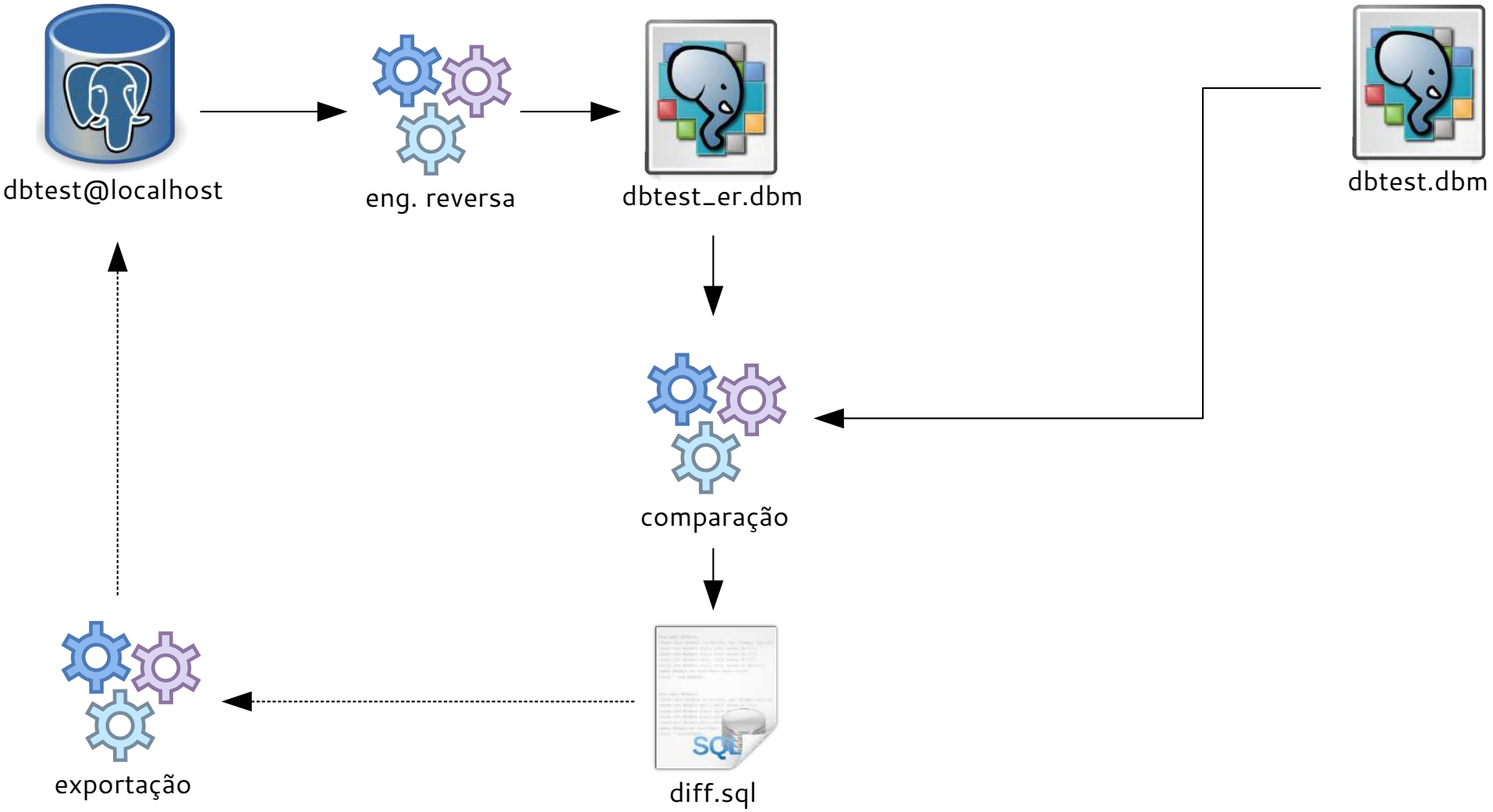
Comparação de banco de dados



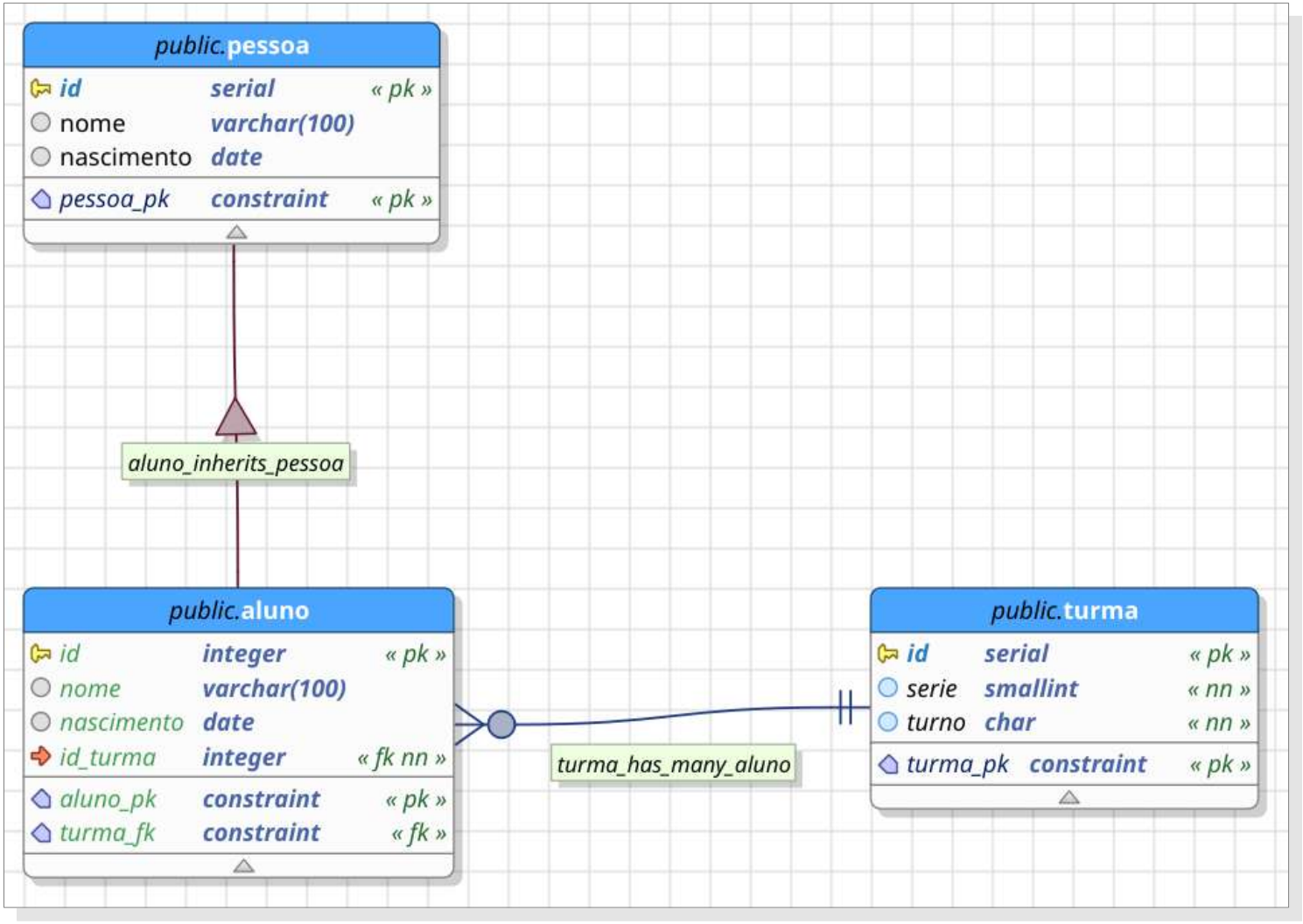
Comparação de banco de dados



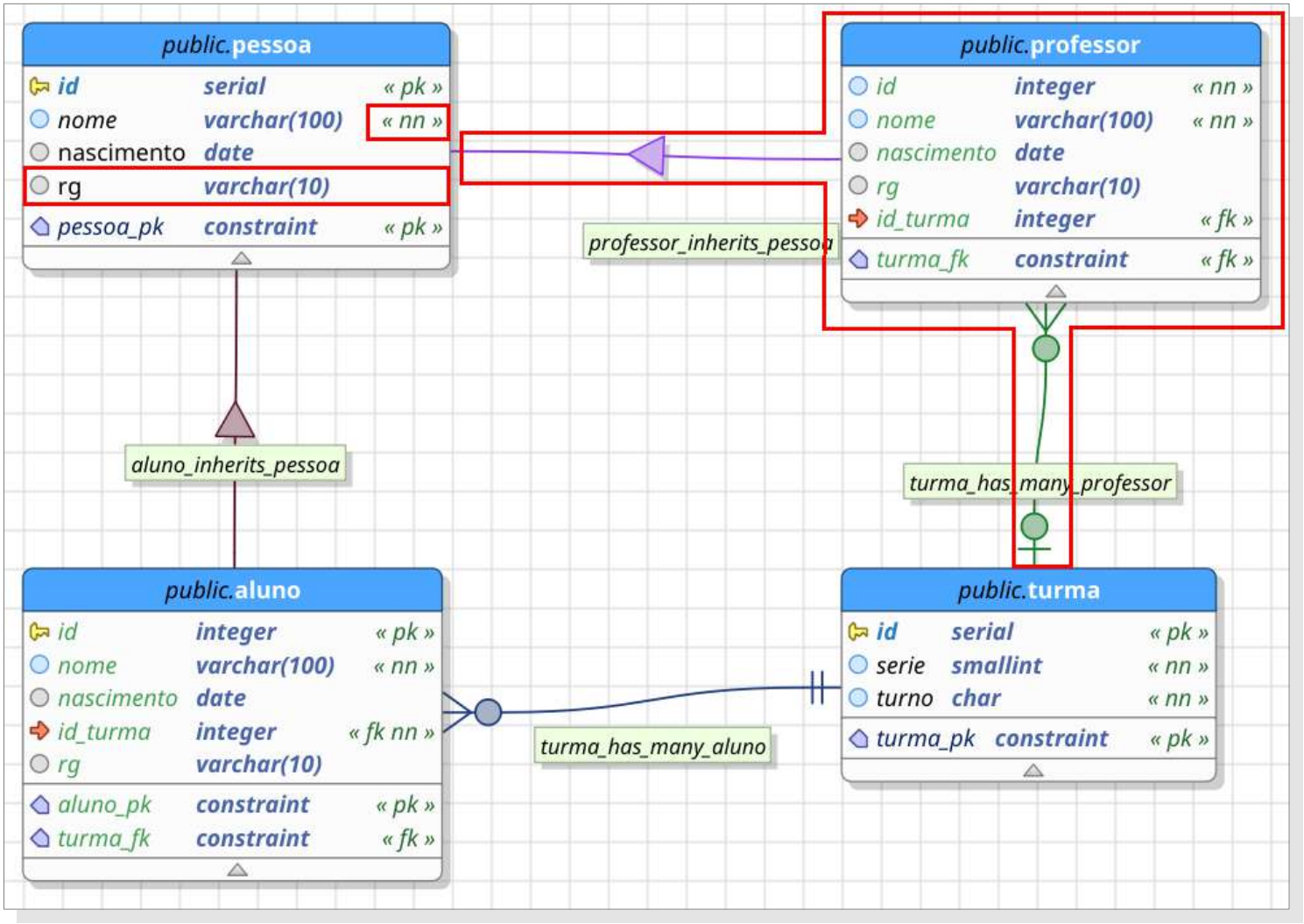
Comparação de banco de dados



Comparação de banco de dados



Comparação de banco de dados

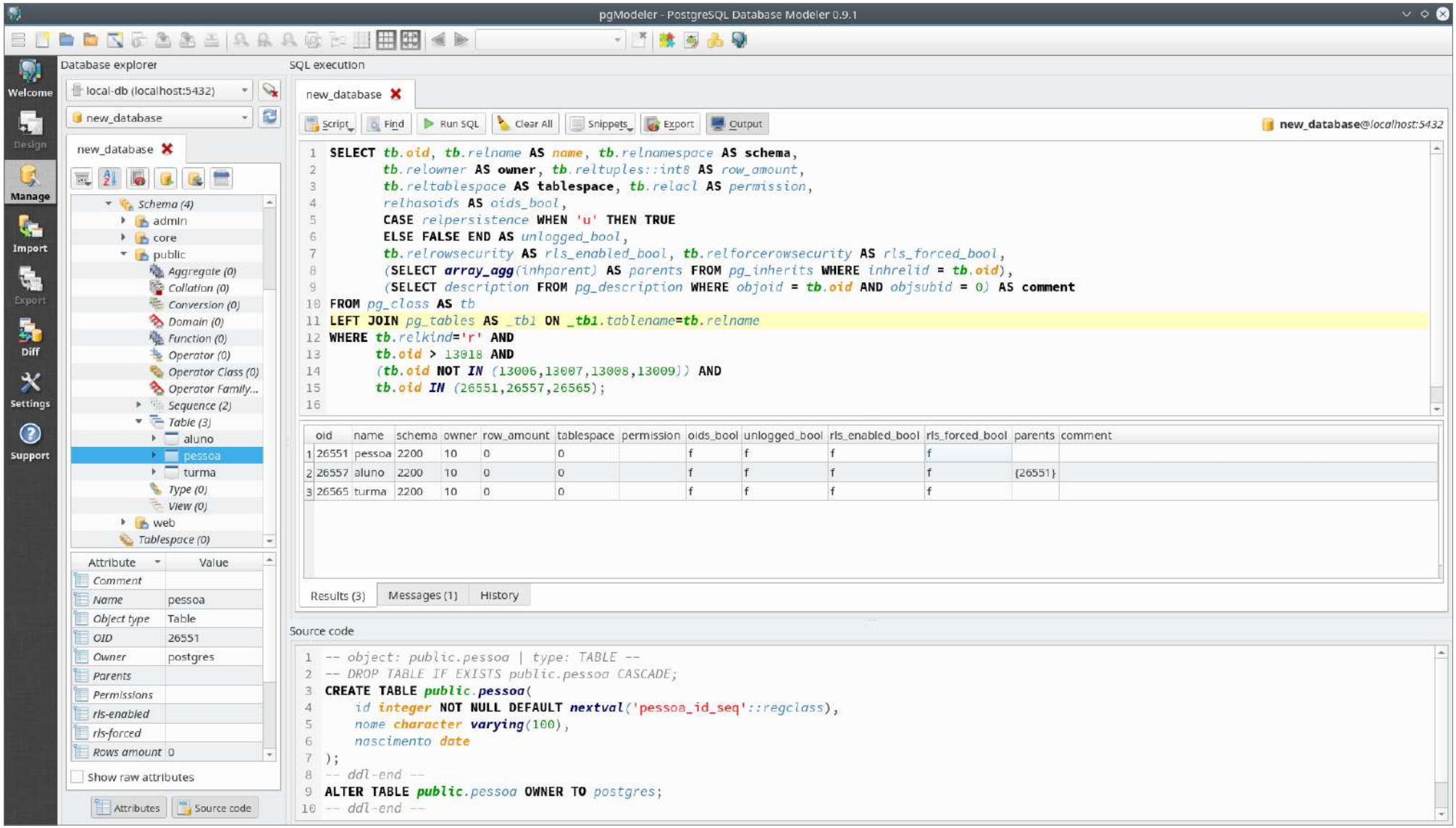


Comparação de banco de dados

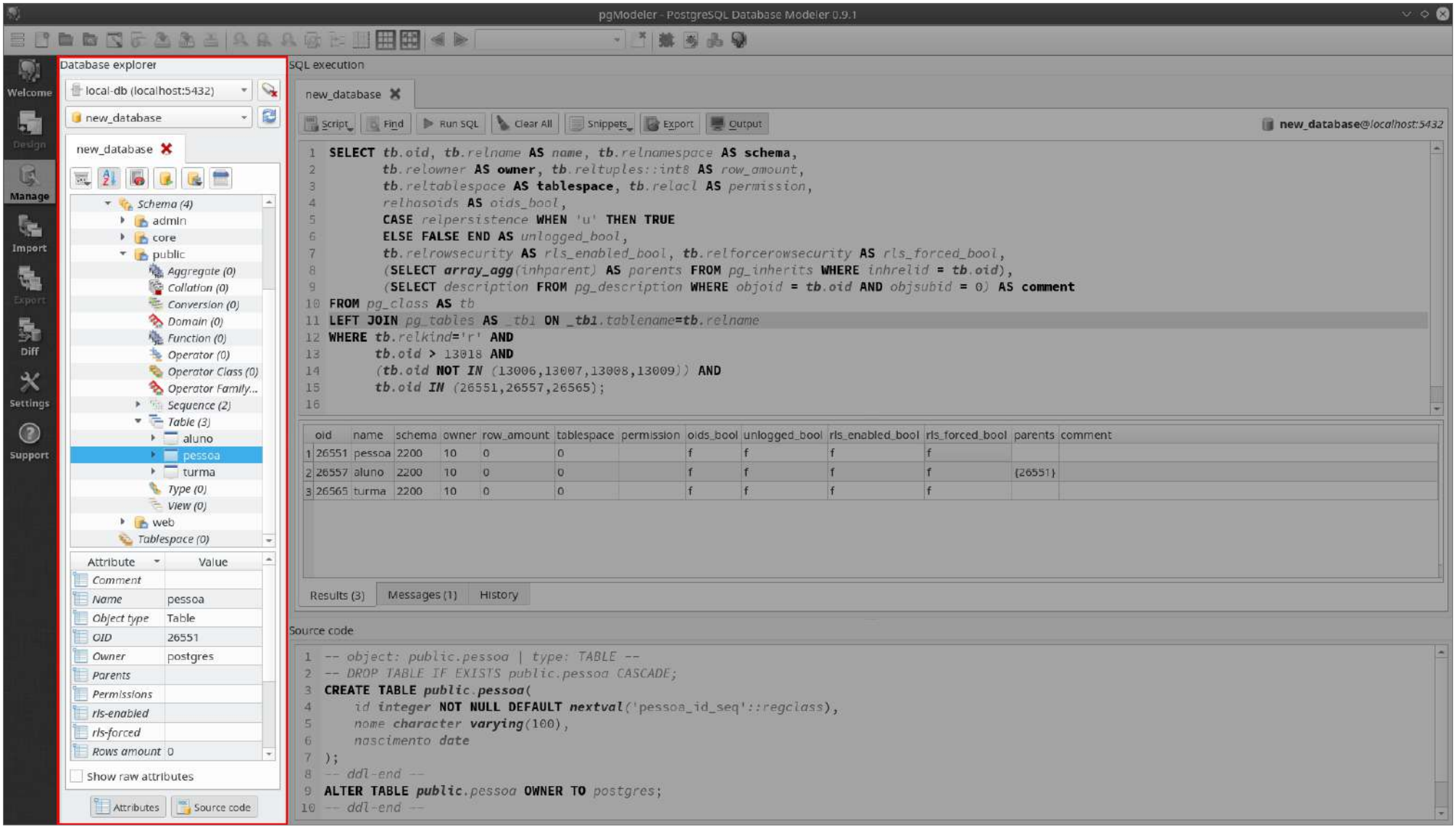
```
1  SET search_path=public,pg_catalog,core,web,admin;
2  -- ddl-end --
3
4  -- [ Created objects ] --
5  -- object: rg | type: COLUMN --
6  -- ALTER TABLE public.pessoa DROP COLUMN IF EXISTS rg CASCADE;
7  ALTER TABLE public.pessoa ADD COLUMN rg varchar(10);
8  -- ddl-end --
9
10 -- object: public.professor | type: TABLE --
11 -- DROP TABLE IF EXISTS public.professor CASCADE;
12 CREATE TABLE public.professor(
13     id_turma integer
14 --  id integer NOT NULL,
15 --  nome varchar(100) NOT NULL,
16 --  nascimento date,
17 --  rg varchar(10),
18
19 ) INHERITS(public.pessoa)
20 ;
21 -- ddl-end --
22 ALTER TABLE public.professor OWNER TO postgres;
23 -- ddl-end --
24
25 -- [ Changed objects ] --
26 ALTER TABLE public.pessoa ALTER COLUMN nome SET NOT NULL;
27 -- ddl-end --
28
29 -- [ Created foreign keys ] --
30 -- object: turma_fk | type: CONSTRAINT --
31 -- ALTER TABLE public.professor DROP CONSTRAINT IF EXISTS turma_fk CASCADE;
32 ALTER TABLE public.professor ADD CONSTRAINT turma_fk FOREIGN KEY (id_turma)
33 REFERENCES public.turma (id) MATCH FULL
34 ON DELETE SET NULL ON UPDATE CASCADE;
35 -- ddl-end --
```

Administração de BDs

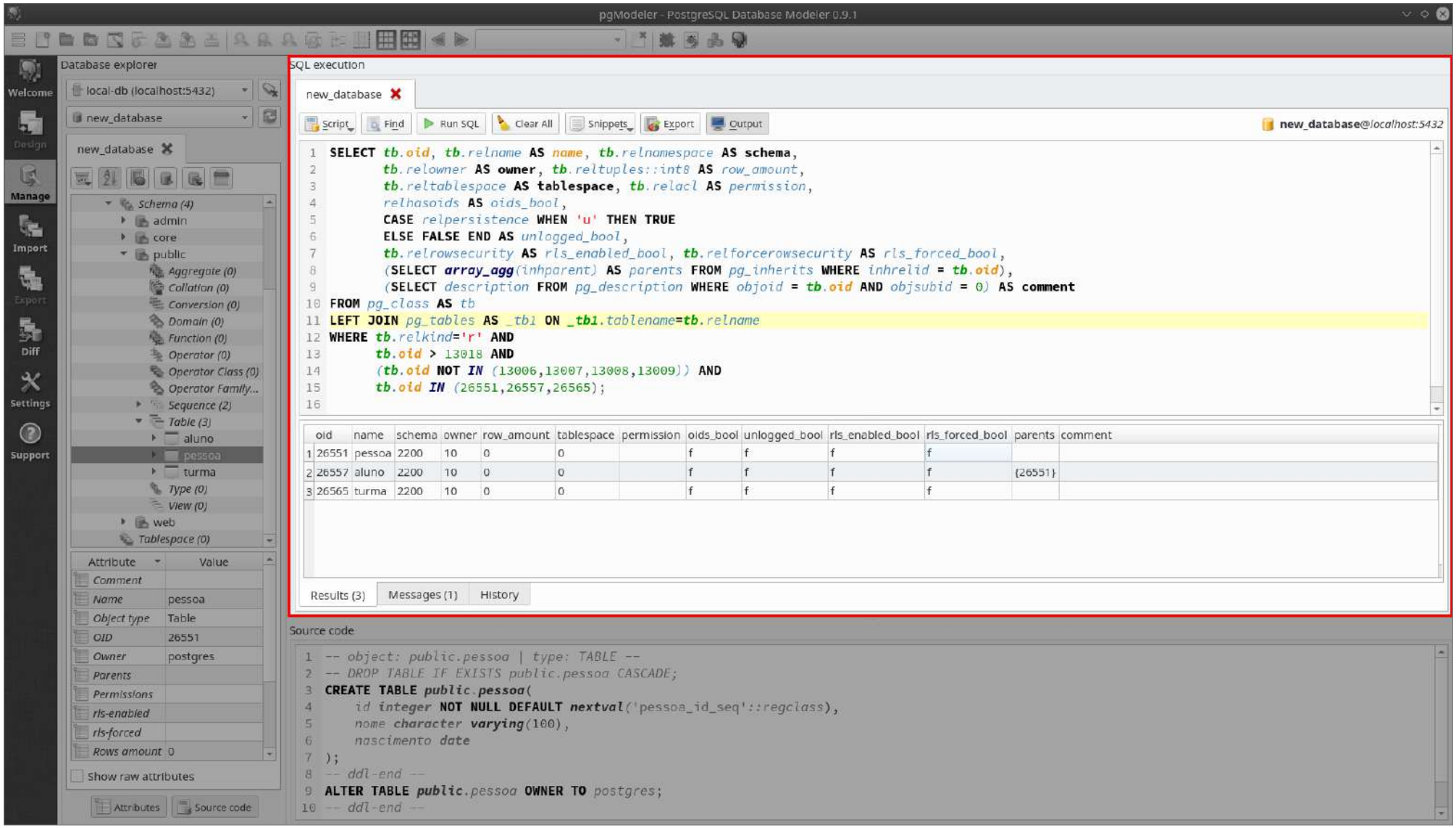
Administração de banco de dados



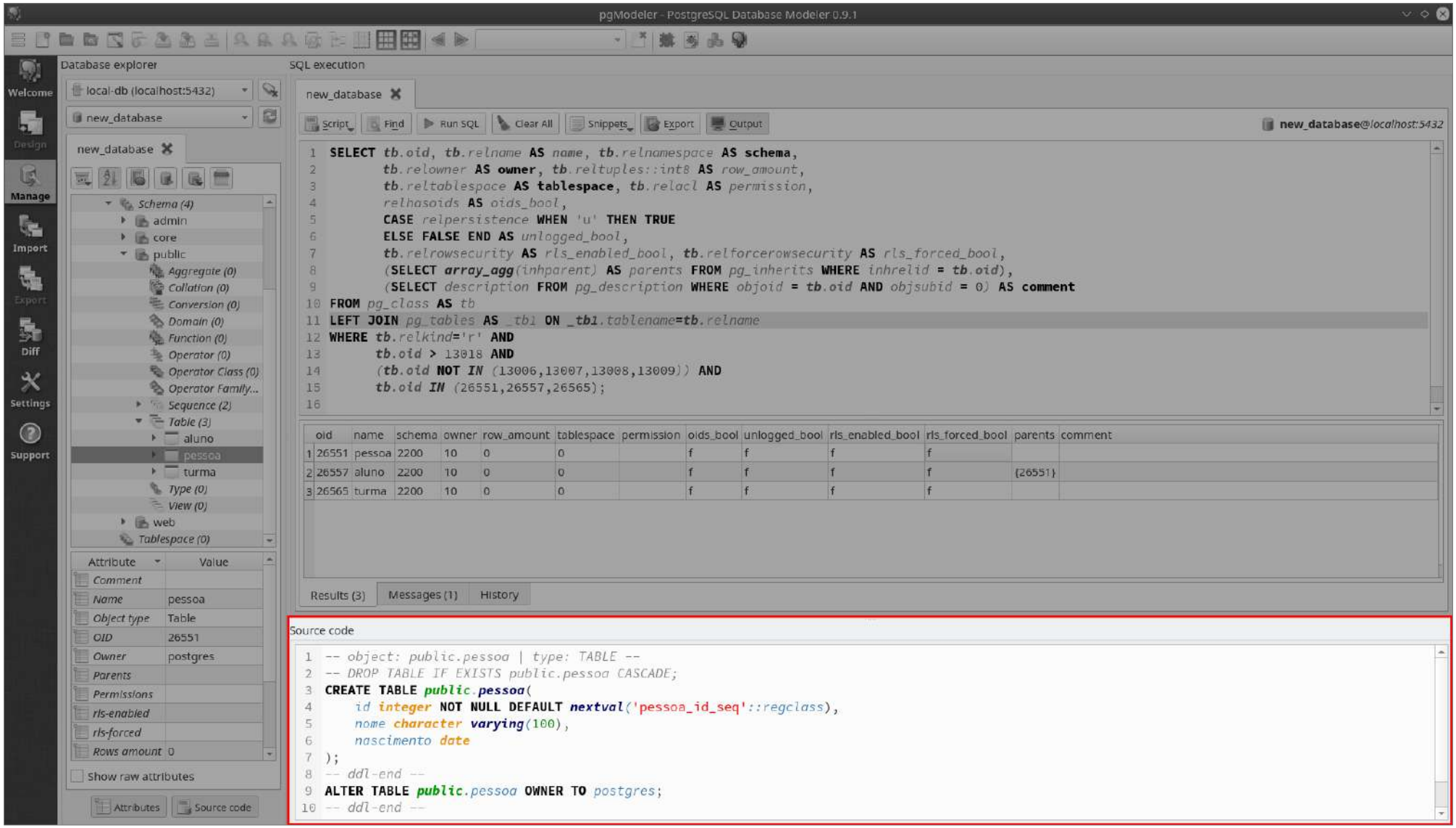
Administração de banco de dados



Administração de banco de dados



Administração de banco de dados

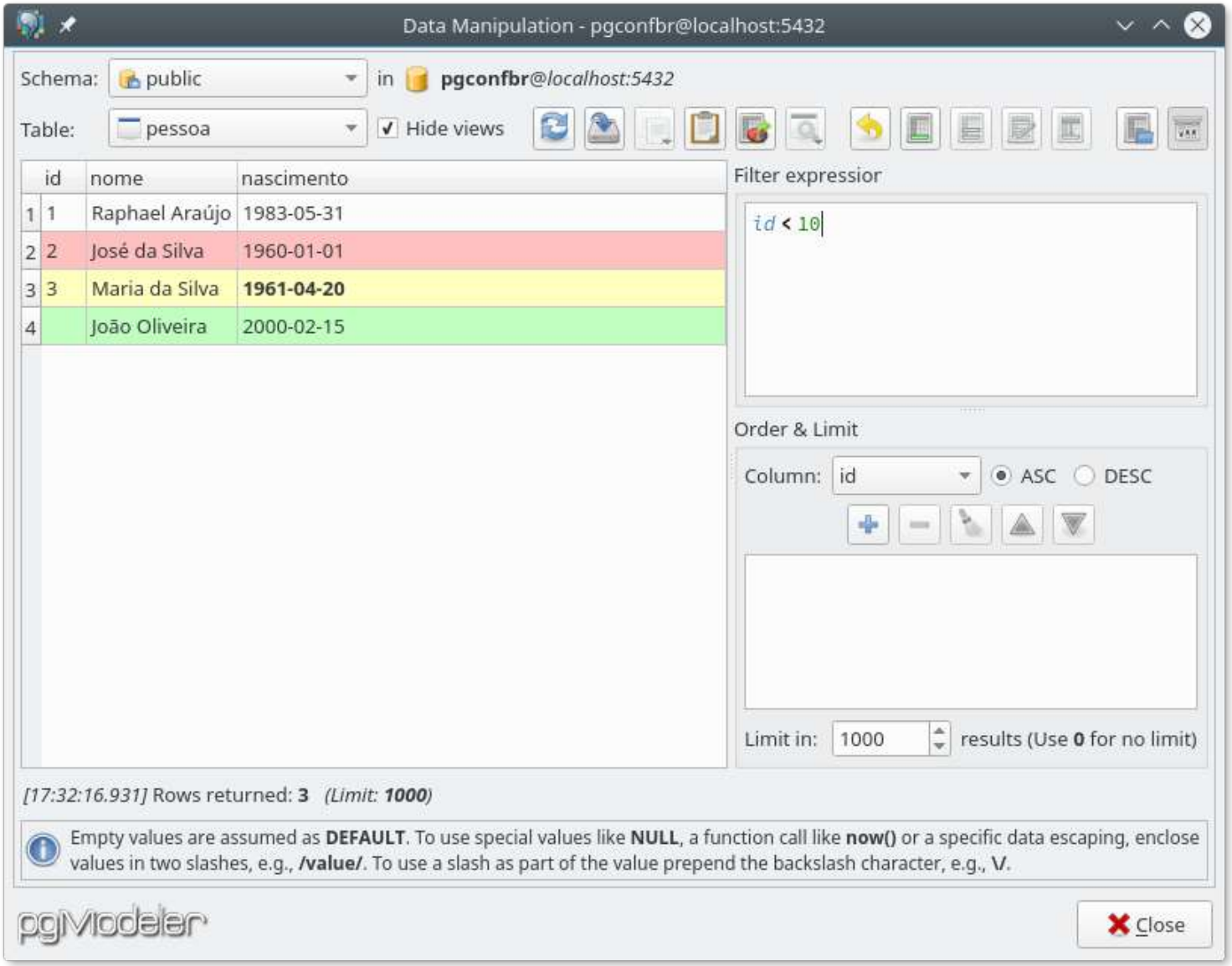


Manipulação de dados

Manipulação de dados

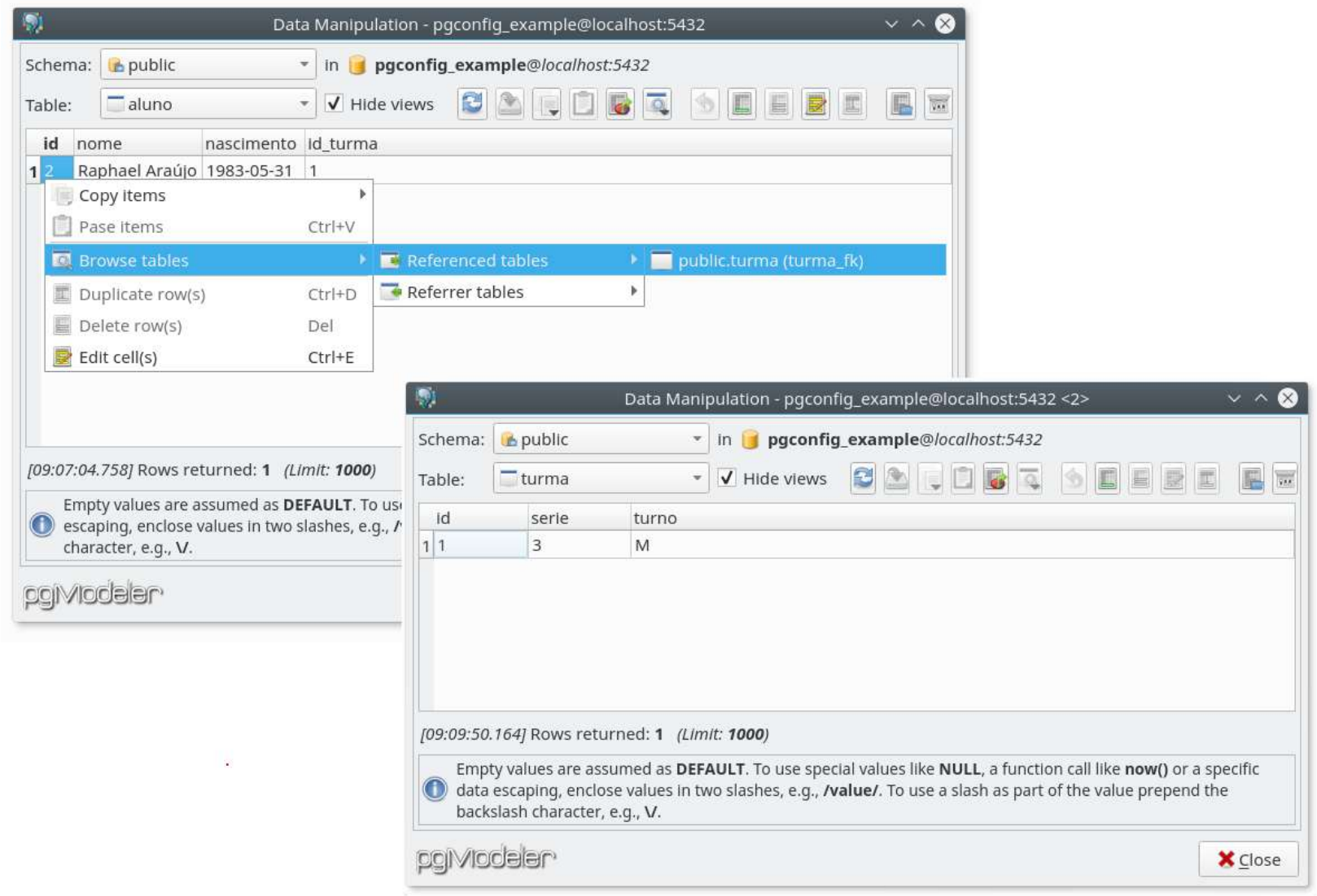
- Feita em formulário dedicado;
- Destaca cada tipo de operação sobre um registro;
- Permite a navegação entre registros que se relacionam;
- Extras: filtragem e ordenação, exportação de resultados para CSV, alimentação de dados via CSV;

Manipulação de dados



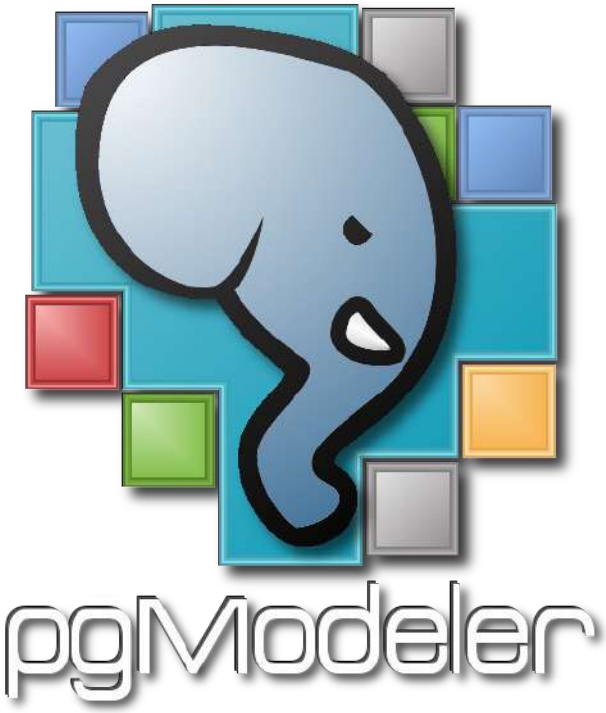
Manipulação de dados

- Navegação entre registros:



E no futuro?

- Melhorias na interface e usabilidade;
- Suporte ao particionamento de tabelas;
- Suporte a criação de objetos através formulários no módulo de administração;
- Engenharia reversa de bancos de dados MySQL / MariaDB;
- ...e muitas outras!



Muito obrigado!

https://pgmodeler.io

raphael@pgmodeler.io

/rkhaotix /raphasilva /pgmodeler /pgmodeler